SQL *from Apps*

Week 4, Day 2

Objective

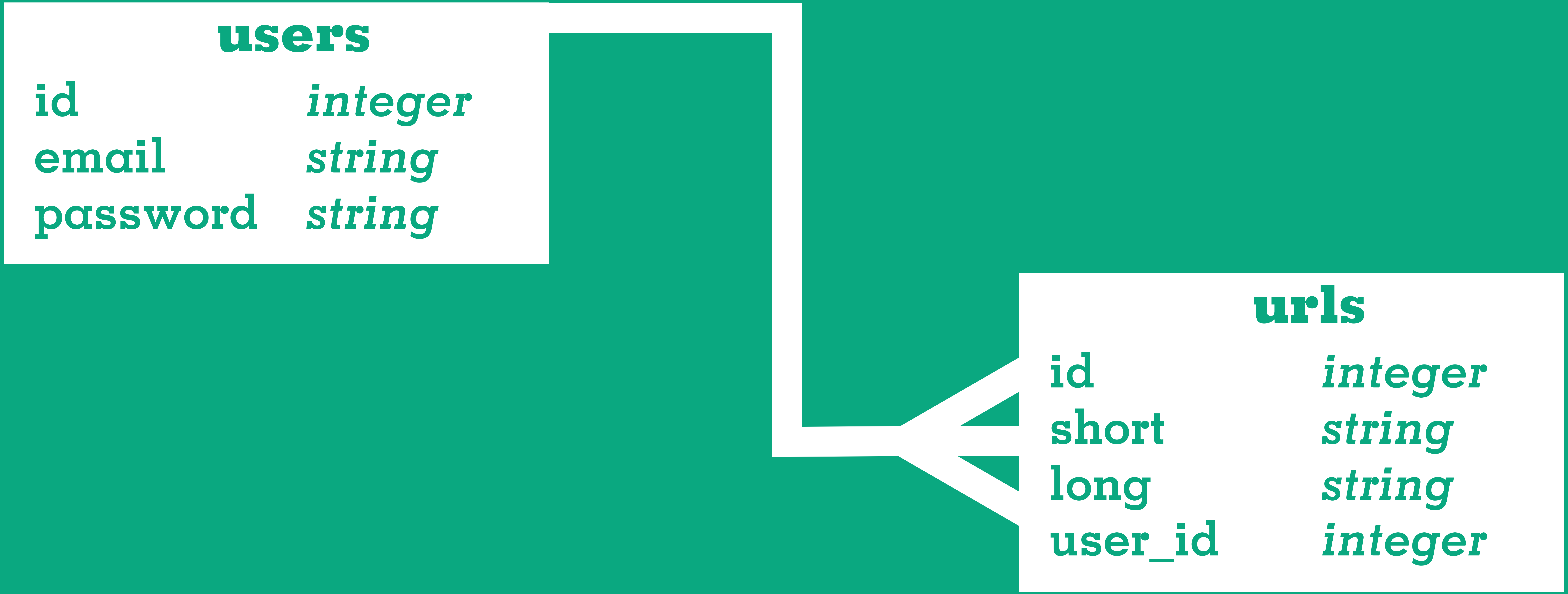
- Reading & Writing data using SQL
- Executing queries asynchronously
- Ensuring secure user input

TinyApp

The data stored by TinyApp was in memory.

There was no persistence of data.

A ‘**user**’ has many ‘**urls**’.



users	
id	*integer*
email	*string*
password	*string*

urls	
id	*integer*
short	*string*
long	*string*
user_id	*integer*

Import Schema

```
\i tinyapp.sql
```

in PostgreSQL

Reading & Writing data using SQL

GET /login

```
SELECT *
FROM users;
```

GET /urls

```sql
SELECT short, long
FROM urls
WHERE user_id = 1;
```

POST /urls

```
INSERT INTO urls (short, long, user_id)
VALUES ('abc', ‘http://www.ask.com/', 1);
```

GET /urls/:short

```
SELECT short, long
FROM urls
WHERE short = 'abc';
```

POST /urls/:short/edit

```
UPDATE urls
SET long = ‘http://www.ask.com/'
WHERE short = 'abc';
```

POST /urls/:short/delete

```
DELETE
FROM urls
WHERE short = 'abc';
```

Executing queries…

…**async**hronously

```
npm install pg —save
```

postgres w/

JavaScript

vs

Ruby

const **using** =
require(‘**modules**’);

Using Modules

exported.js

```
module.exports = () => {
 console.log(‘Imported!’);
}
```

```
const imported = require(‘./exported’);

imported();
```

Secure

User Input

The ‘**pg**’ library provides a way to escape user input.

```
$1::text
$2::integer
```